

QUESTION 1:

1.customer-wise total spending and number of visits (Each order/sale done by a customer can be considered as a visit).

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab contains the following SQL query:

```
SELECT s.customer_code,COUNT(*) AS num_visits,
       SUM(s.price) AS total_spent
  FROM sales s
  JOIN menu m ON s.product_id = m.product_id
 GROUP BY s.customer_code
 ORDER BY total_spent DESC;
```

The 'Script Output' tab shows the results:

CUSTO	NUM_VISITS	TOTAL_SPENT
A1	6	57.52
B2	6	55.66
C3	3	38.55

2. For each customer, show the first item name and date on which such item was purchased.

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab contains the following SQL query:

```
SELECT s.customer_code,m.product_name,
       TO_CHAR(s.order_date, 'YYYY-MM-DD') AS first_purchase_date
  FROM sales s
  JOIN menu m ON s.product_id = m.product_id
 GROUP BY s.customer_code
 ORDER BY total_spent DESC;
```

The 'Script Output' tab shows the results:

CUSTO	PRODU	FIRST_PURC
A1	6	57.52
B2	6	55.66
C3	3	38.55

CUSTO	PRODU	FIRST_PURC
A1	Pizza	2024-03-01
A1	Mazza	2024-03-01
B2	Ramen	2024-02-01
C3	Ramen	2024-03-01
C3	Ramen	2024-03-01

3. Top 5 Most Frequently Purchased item (Overall).

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab contains the following SQL code:

```
SELECT m.product_id, m.product_name,
       COUNT(*) AS times_sold
  FROM sales s
 JOIN menu m ON s.product_id = m.product_id
 GROUP BY m.product_id, m.product_name
 ORDER BY times_sold DESC
FETCH FIRST 5 ROWS ONLY;
```

The 'Script Output' tab shows the results:

PRODUCT_ID	PRODU	TIMES SOLD
3	Ramen	8
2	Mazza	4
1	Pizza	3

4. Customer-wise, the most frequently purchased item.

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab contains the following SQL code:

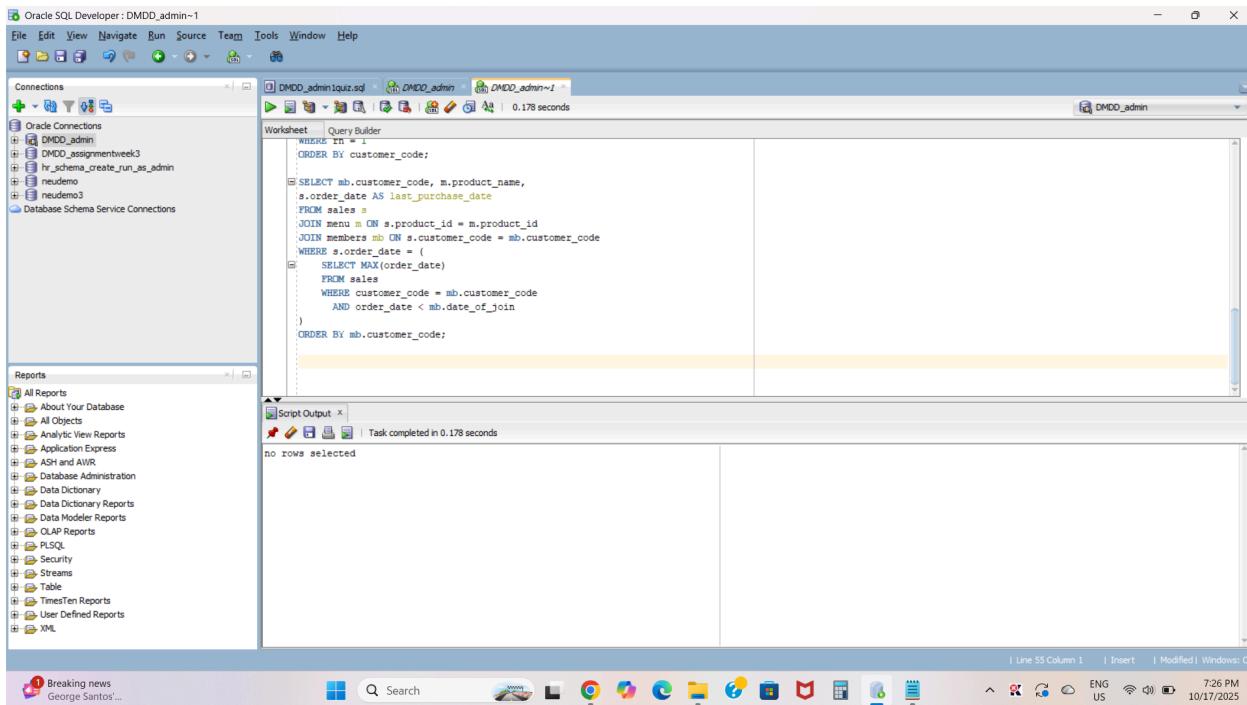
```
GROUP BY m.product_id, m.product_name
ORDER BY times_sold DESC
FETCH FIRST 5 ROWS ONLY;

WITH item_counts AS (
  SELECT s.customer_code, s.product_id, m.product_name, COUNT(*) AS cnt
  FROM sales s
  JOIN menu m ON s.product_id = m.product_id
  GROUP BY s.customer_code, s.product_id, m.product_name
)
SELECT customer_code, product_id, product_name, cnt
FROM (
  SELECT ic.*, ROW_NUMBER() OVER (PARTITION BY customer_code ORDER BY cnt DESC) AS rn
  FROM item_counts ic
)
WHERE rn = 1
ORDER BY customer_code;
```

The 'Script Output' tab shows the results:

CUSTO	PRODUCT_ID	PRODU	CNT
A1	3	Ramen	3
B2	2	Mazza	2
C3	3	Ramen	3

5.Customer-wise show the last item that was purchased before becoming a member.



The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar lists several databases, including 'DMDD_admin', 'DMDD_assignmentweek3', 'hr_schema_create_run_as_admin', 'neudemo', and 'neudemo3'. The 'Reports' sidebar contains various report categories. The 'Worksheet' tab displays a query:

```

WHERE rn = 1
ORDER BY customer_code;

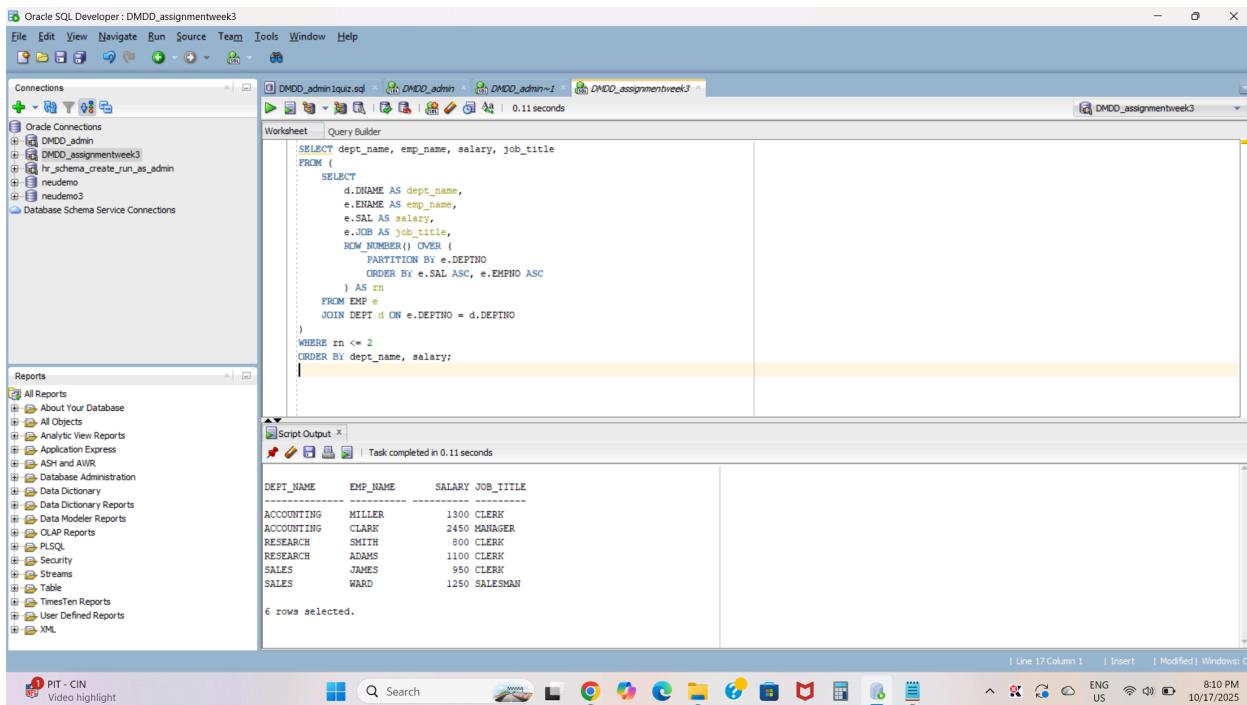
SELECT mb.customer_code, m.product_name,
       s.order_date AS last_purchase_date
  FROM sales s
 JOIN menu m ON s.product_id = m.product_id
 JOIN members mb ON s.customer_code = mb.customer_code
 WHERE s.order_date = (
   SELECT MAX(order_date)
     FROM sales
    WHERE customer_code = mb.customer_code
      AND order_date < mb.date_of_join
)
ORDER BY mb.customer_code;

```

The 'Script Output' tab shows the result: "no rows selected". The status bar at the bottom right indicates the task completed in 0.178 seconds.

QUESTION 2:

- Find the top two least paid employees in each department, along with their department name, employee name, salary, job title.



The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar lists databases: 'DMDD_admin', 'DMDD_assignmentweek3', 'hr_schema_create_run_as_admin', 'neudemo', and 'neudemo3'. The 'Reports' sidebar is visible. The 'Worksheet' tab displays a query:

```

SELECT dept_name, emp_name, salary, job_title
  FROM (
    SELECT
      d.DNAME AS dept_name,
      e.ENAME AS emp_name,
      e.SAL AS salary,
      e.JOB AS job_title,
      ROW_NUMBER() OVER (
        PARTITION BY e.DEPTNO
        ORDER BY e.SAL ASC, e.EMPNO ASC
      ) AS rn
     FROM EMP e
    JOIN DEPT d ON e.DEPTNO = d.DEPTNO
  )
 WHERE rn <= 2
 ORDER BY dept_name, salary;

```

The 'Script Output' tab shows the results:

DEPT_NAME	EMP_NAME	SALARY	JOB_TITLE
ACCOUNTING	MILLER	1300	CLERK
ACCOUNTING	CLARK	2450	MANAGER
RESEARCH	SMITH	800	CLERK
RESEARCH	ADAMS	1100	CLERK
SALES	JAMES	950	CLERK
SALES	WARD	1250	SALESMAN

6 rows selected.

The status bar at the bottom right indicates the task completed in 0.11 seconds.

2.List employees who have a salary greater than the average salary of all employees.

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab contains the following SQL code:

```
SELECT
    d.DNAME AS dept_name,
    e.ENAME AS emp_name,
    e.SAL AS salary,
    e.JOB AS job_title,
    ROWNUMBER() OVER (
        PARTITION BY e.DEPTNO
        ORDER BY e.SAL ASC, e.ENMNO ASC
    ) AS rn
FROM EMP e
JOIN DEPT d ON e.DEPTNO = d.DEPTNO
)
WHERE rn <= 2
ORDER BY dept_name, salary;

SELECT ENAME, JOB, SAL, DEPTNO
FROM EMP
WHERE SAL > (SELECT AVG(SAL) FROM EMP)
ORDER BY SAL DESC;
```

The 'Script Output' tab shows the results of the query:

ENAME	JOB	SAL	DEPTNO
KING	PRESIDENT	5000	10
FORD	ANALYST	3000	20
SCOTT	ANALYST	3000	20
JONES	MANAGER	2975	20
BLAKE	MANAGER	2850	30
CLARK	MANAGER	2450	10

6 rows selected.

3.List employees who were hired in the same month and year as the employee SCOTT.

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab contains the following SQL code:

```
SELECT
    d.DNAME AS dept_name,
    e.ENAME AS emp_name,
    e.SAL AS salary,
    e.JOB AS job_title,
    ROWNUMBER() OVER (
        PARTITION BY e.DEPTNO
        ORDER BY e.SAL ASC, e.ENMNO ASC
    ) AS rn
FROM EMP e
JOIN DEPT d ON e.DEPTNO = d.DEPTNO
)
WHERE rn <= 2
ORDER BY dept_name, salary;

SELECT ENAME, JOB, SAL, DEPTNO
FROM EMP
WHERE SAL > (SELECT AVG(SAL) FROM EMP)
ORDER BY SAL DESC;

SELECT ENAME, JOB, HIREDATE, DEPTNO, SAL
FROM EMP
WHERE TO_CHAR(HIREDATE, 'MM-YYYY') =
    (SELECT TO_CHAR(HIREDATE, 'MM-YYYY')
    FROM EMP
    WHERE ENAME = 'SCOTT');
```

The 'Script Output' tab shows the results of the query:

ENAME	JOB	HIREDATE	DEPTNO	SAL
SCOTT	ANALYST	3000	20	
JONES	MANAGER	2975	20	
BLAKE	MANAGER	2850	30	
CLARK	MANAGER	2450	10	

6 rows selected.

ENAME	JOB	HIREDATE	DEPTNO	SAL
SCOTT	ANALYST	09-DEC-82	20	3000

4. List employee details with a salary greater than the average salary in their department.

```

File Edit View Navigate Run Source Team Tools Window Help
File Edit View Navigate Run Source Team Tools Window Help
Connections DMDD_admin DMDD_assignmentweek3 DMDD_admin DMDD_assignmentweek3
DMDD_assignmentweek3 hr_schema_create_run_as_admin neudemo neudemo3 Database Schema Service Connections
Reports All Reports About Your Database All Objects Analytic View Reports Application Express ASH and AWR Database Administration Data Dictionary Data Dictionary Reports Data Modeler Reports OLAP Reports PLSQL Security Streams Table TimesTen Reports User Defined Reports XML
Script Output X Task completed in 0.671 seconds
WHERE SAL > (SELECT AVG(SAL) FROM EMP)
ORDER BY SAL DESC;

SELECT ENAME, JOB, HIREDATE, DEPTNO, SAL
FROM EMP
WHERE TO_CHAR(HIREDATE, 'MM-YYYY') =
(SELECT TO_CHAR(HIREDATE, 'MM-YYYY')
FROM EMP
WHERE ENAME = 'SCOTT');

SELECT e.ENAME, e.JOB, e.SAL, d.DNAME AS DEPT_NAME
FROM EMP e
JOIN DEPT d ON e.DEPTNO = d.DEPTNO
WHERE e.SAL > (
    SELECT AVG(SAL)
    FROM EMP
    WHERE DEPTNO = e.DEPTNO
)
ORDER BY d.DNAME, e.SAL DESC;

```

ENAME	JOB	SAL	DEPT_NAME
KING	PRESIDENT	5000	ACCOUNTING
FORD	ANALYST	3000	RESEARCH
SCOTT	ANALYST	3000	RESEARCH
JONES	MANAGER	2975	RESEARCH
BLAKE	MANAGER	2850	SALES
ALLEN	SALESMAN	1600	SALES

6 rows selected.

5. Which departments have the maximum and least number of employees.

```

File Edit View Navigate Run Source Team Tools Window Help
File Edit View Navigate Run Source Team Tools Window Help
Connections DMDD_admin DMDD_assignmentweek3 DMDD_admin DMDD_assignmentweek3
DMDD_assignmentweek3 hr_schema_create_run_as_admin neudemo neudemo3 Database Schema Service Connections
Reports All Reports About Your Database All Objects Analytic View Reports Application Express ASH and AWR Database Administration Data Dictionary Data Dictionary Reports Data Modeler Reports OLAP Reports PLSQL Security Streams Table TimesTen Reports User Defined Reports XML
Script Output X Task completed in 0.8190001 seconds
SELECT e.ENAME, e.JOB, e.SAL, d.DNAME AS DEPT_NAME
FROM EMP e
JOIN DEPT d ON e.DEPTNO = d.DEPTNO
WHERE e.SAL > (
    SELECT AVG(SAL)
    FROM EMP
    WHERE DEPTNO = e.DEPTNO
)
ORDER BY d.DNAME, e.SAL DESC;

SELECT d.DEPTNO, d.DNAME, COUNT(e.EMPNO) AS emp_count
FROM DEPT d
LEFT JOIN EMP e ON d.DEPTNO = e.DEPTNO
GROUP BY d.DEPTNO, d.DNAME
HAVING COUNT(e.EMPNO) = (SELECT MAX(cnt) FROM (SELECT COUNT(*) AS cnt FROM EMP GROUP BY DEPTNO))
    OR COUNT(e.EMPNO) = (SELECT MIN(cnt) FROM (SELECT COUNT(*) AS cnt FROM EMP GROUP BY DEPTNO))
ORDER BY emp_count DESC;

```

DEPTNO	DNAME	EMP_COUNT
30	SALES	6
10	ACCOUNTING	3

6 rows selected.

QUESTION 3:

1. Insert from multiple sessions should continue without any wait so that all inserts are successful for a given employee number to EMP table. Simulate the scenario on your SQL Developer and upload screen shots and scripts.

STEP 1: CREATE USERS .

The screenshot shows the Oracle SQL Developer interface. In the Worksheet tab, the following SQL script is run:

```
CREATE USER DEMO1 IDENTIFIED BY "Matchelatte823";
CREATE USER DEMO2 IDENTIFIED BY "Matchelatte806";

GRANT CONNECT, RESOURCE TO DEMO1;
GRANT CONNECT, RESOURCE TO DEMO2;
```

In the Script Output window, the results are displayed:

```
Grant succeeded.  
User DEMO1 created.  
User DEMO2 created.  
Grant succeeded.  
Grant succeeded.
```

STEP 2: CREATING TABLE.

The screenshot shows the Oracle SQL Developer interface. In the Worksheet tab, the following SQL script is run:

```
DROP TABLE EMP CASCADE CONSTRAINTS;

CREATE TABLE EMP (
    EMPNO NUMBER PRIMARY KEY,
    ENAME VARCHAR2(50),
    JOB VARCHAR2(50),
    SAL NUMBER
);

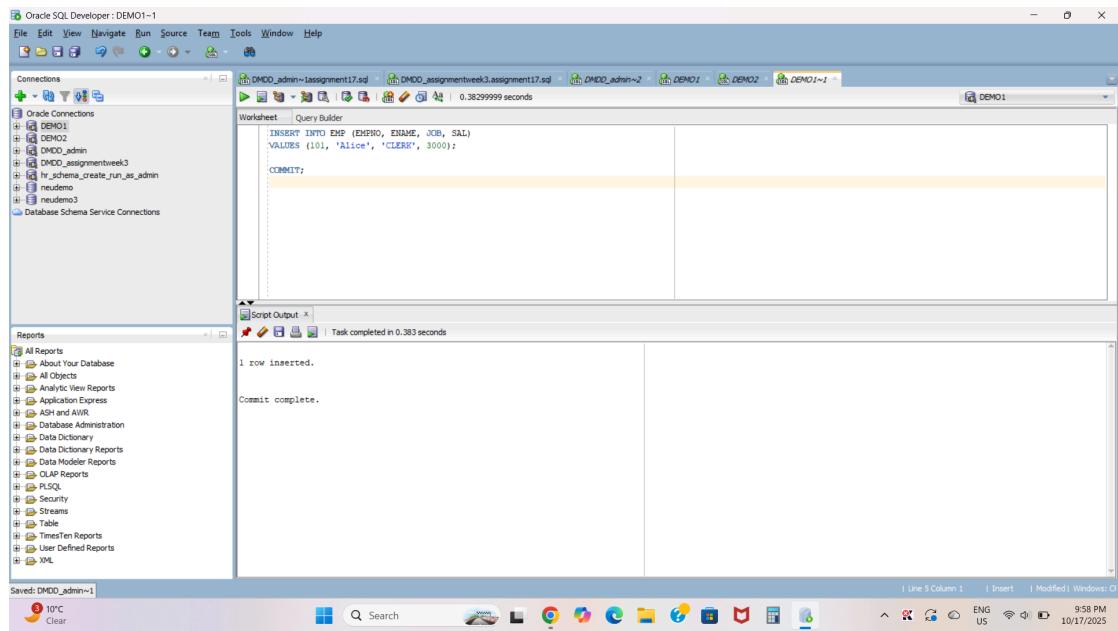
GRANT INSERT, SELECT ON EMP TO DEMO2;
```

In the Script Output window, the results are displayed:

```
Table EMP created.  
Table EMP dropped.  
Table EMP created.  
Grant succeeded.
```

STEP 3:INSERTING DATA

DEMO 1:



The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar lists several databases, including DEMO1, DEMO2, and DEMO_admin. The 'Worksheet' tab displays a query in the 'Query Builder' pane:

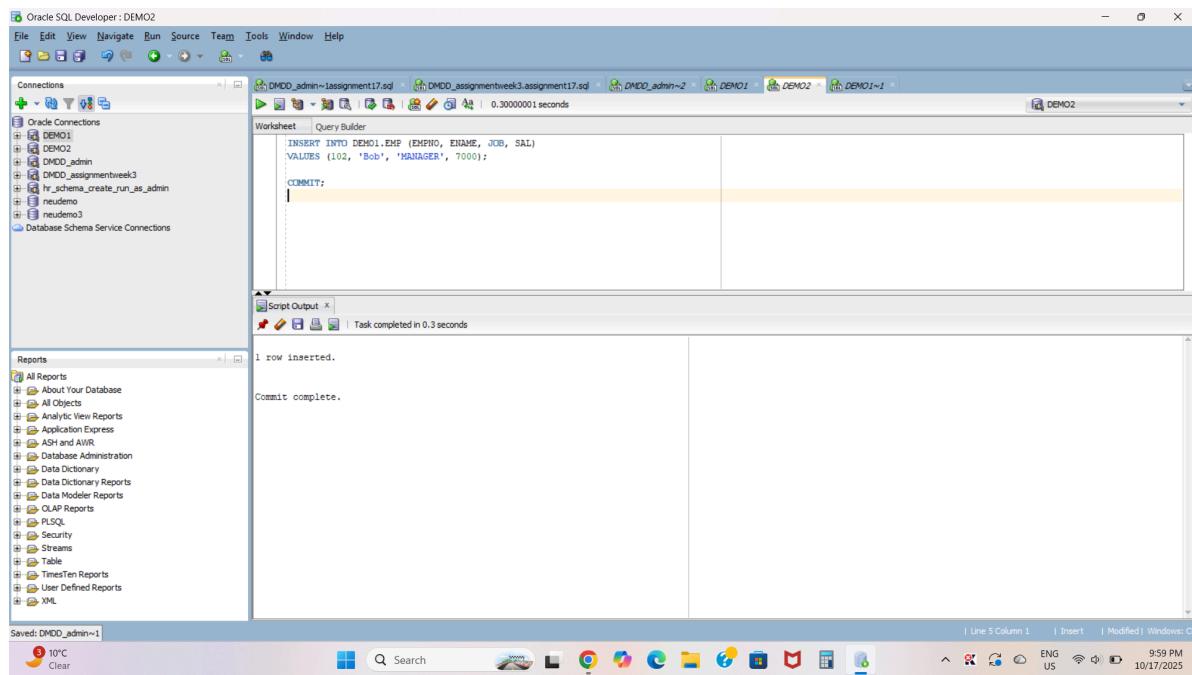
```
INSERT INTO EMP (EMNO, ENAME, JOB, SAL)
VALUES (101, 'Alice', 'CLERK', 3000);
COMMIT;
```

The 'Script Output' pane at the bottom shows the results of the execution:

```
1 row inserted.  
Commit complete.
```

The status bar at the bottom right indicates the task completed in 0.3829999 seconds.

DEMO 2:



The screenshot shows the Oracle SQL Developer interface. The 'Connections' sidebar lists databases DEMO1, DEMO2, and DEMO_admin. The 'Worksheet' tab displays a query in the 'Query Builder' pane:

```
INSERT INTO DEMO1.EMP (EMNO, ENAME, JOB, SAL)
VALUES (102, 'Bob', 'MANAGER', 7000);
COMMIT;
```

The 'Script Output' pane at the bottom shows the results of the execution:

```
1 row inserted.  
Commit complete.
```

The status bar at the bottom right indicates the task completed in 0.3000000 seconds.

STEP 4: OUTPUT OF THE TABLE:

The screenshot shows the Oracle SQL Developer interface. In the 'Worksheet' tab, a script is run:

```
INSERT INTO EMP (EMPNO, ENAME, JOB, SAL)
VALUES (101, 'Alice', 'CLERK', 3000);

COMMIT;

SELECT * FROM EMP;
```

The 'Script Output' tab shows the results:

```
1 row inserted.

Commit complete.
```

EMPNO	ENAME	JOB	SAL
102	Bob	MANAGER	7000
101	Alice	CLERK	3000

2. Identify the different constraint types and their usage for the following:

- O/V:Check Option on View
Prevents inserting or updating rows in a **view** if they don't match the view's filter.
- R/F:Foreign Key
Ensures a column value **exists in another table**.
- U:Unique
Ensures **no duplicate values** in a column.
- C:Check
Column values must meet a rule.
- P:Primary Key
Unique ID for each row and it cannot be NULL.

3. Write an SQL statement which will generate CREATE TABLE script (you can ignore constraints that are not part of the user tab cols).

```

SELECT d.DEPNNO, d.DNAME, COUNT(e.EMPNO) AS emp_count
FROM DEPT d
LEFT JOIN EMP e ON d.DEPNNO = e.DEPNNO
GROUP BY d.DEPNNO, d.DNAME
HAVING COUNT(e.EMPNO) = (SELECT MAX(cnt) FROM (SELECT COUNT(*) AS cnt FROM EMP GROUP BY DEPNTNO))
OR COUNT(e.EMPNO) = (SELECT MIN(cnt) FROM (SELECT COUNT(*) AS cnt FROM EMP GROUP BY DEPNTNO))
ORDER BY emp_count DESC;

SELECT COLUMN_NAME || ' ' || DATA_TYPE || 
CASE
    WHEN DATA_TYPE LIKE 'VARCHAR' THEN '(' || DATA_LENGTH || ')'
    WHEN DATA_TYPE = 'NUMBER' AND DATA_PRECISION IS NOT NULL
        THEN '(' || DATA_PRECISION || ',' || NVL(DATA_SCALE,0) || ')'
    ELSE ''
END AS column_definition
FROM USER_TAB_COLUMNS
WHERE TABLE_NAME = 'EMP'
ORDER BY COLUMN_ID;

```

4. Find an alternate (Other than the option explained in class) method to update the hire date year from 1981 to 2000 for the EMP table row.

```

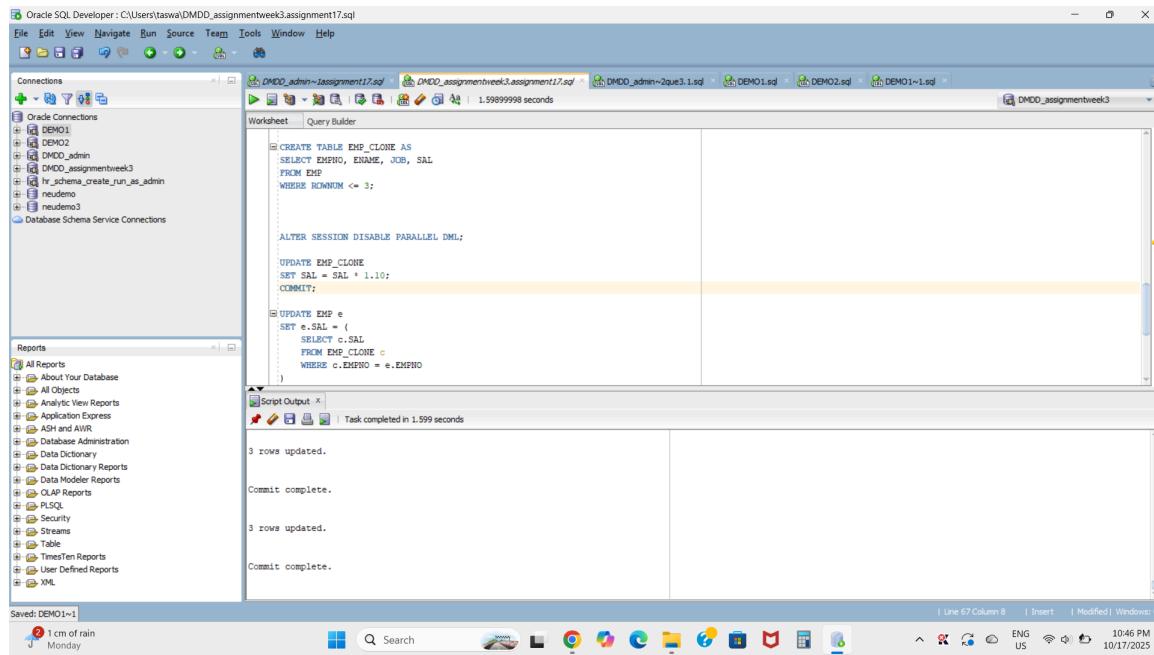
SELECT d.DEPNNO, d.DNAME, COUNT(e.EMPNO) AS emp_count
FROM DEPT d
LEFT JOIN EMP e ON d.DEPNNO = e.DEPNNO
GROUP BY d.DEPNNO, d.DNAME
HAVING COUNT(e.EMPNO) = (SELECT MAX(cnt) FROM (SELECT COUNT(*) AS cnt FROM EMP GROUP BY DEPNTNO))
OR COUNT(e.EMPNO) = (SELECT MIN(cnt) FROM (SELECT COUNT(*) AS cnt FROM EMP GROUP BY DEPNTNO))
ORDER BY emp_count DESC;

UPDATE EMP
SET HIREDATE = ADD_MONTHS(HIREDATE, (2000 - EXTRACT(YEAR FROM HIREDATE)) * 12)
WHERE EXTRACT(YEAR FROM HIREDATE) = 1981;

COMMIT;

```

5. Write an UPDATE DML statement to update the salaries of employees based on another EMP table
 (create a clone of emp with limited set of rows then update the salary and reflect the changes to the master emp table)



```

CREATE TABLE EMP_CLONE AS
SELECT EMPNO, ENAME, JOB, SAL
FROM EMP
WHERE ROWNUM <= 3;

ALTER SESSION DISABLE PARALLEL DML;

UPDATE EMP_CLONE
SET SAL = SAL * 1.10;
COMMIT;

UPDATE EMP e
SET e.SAL = (
  SELECT c.SAL
  FROM EMP_CLONE c
  WHERE c.EMPNO = e.EMPNO
)
  
```

Script Output:

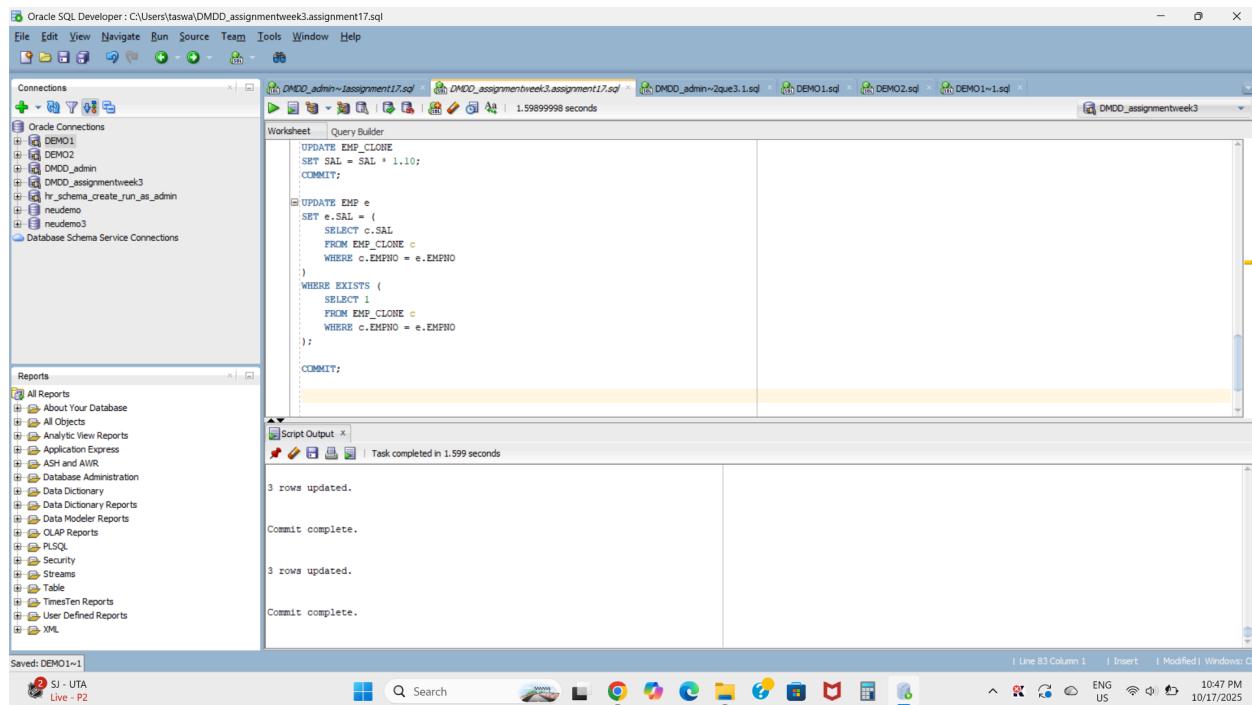
```

3 rows updated.

Commit complete.

3 rows updated.

Commit complete.
  
```



```

UPDATE EMP_CLONE
SET SAL = SAL * 1.10;
COMMIT;

UPDATE EMP e
SET e.SAL = (
  SELECT c.SAL
  FROM EMP_CLONE c
  WHERE c.EMPNO = e.EMPNO
)

WHERE EXISTS (
  SELECT 1
  FROM EMP_CLONE c
  WHERE c.EMPNO = e.EMPNO
);

COMMIT;
  
```

Script Output:

```

3 rows updated.

Commit complete.

3 rows updated.

Commit complete.
  
```

Oracle SQL Developer : C:\Users\taswa\DMDD_assignmentweek3.assignment17.sql

File Edit View Navigate Run Source Team Tools Window Help

Connections

- Oracle Connections
- DEM01
- DEM02
- DMDD_admin
- DMDD_assignmentweek3
- hr_schema_create_run_as_admin
- neudemo
- neudemo3

Database Schema Service Connections

Reports

- All Reports
- About Your Database
- All Objects
- Analytic View Reports
- Application Express
- ASH and AWR
- Database Administration
- Data Dictionary
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- PLSQL
- Security
- Streams
- Table
- TimesTen Reports
- User Defined Reports
- XML

Worksheet Query Builder

```
UPDATE EMP e
SET e.SAL = (
    SELECT c.SAL
    FROM EMP_CLONE c
    WHERE c.EMPNO = e.EMPNO
)
WHERE EXISTS (
    SELECT 1
    FROM EMP_CLONE c
    WHERE c.EMPNO = e.EMPNO
);

COMMIT;

SELECT EMPNO, ENAME, SAL
FROM EMP
ORDER BY EMPNO;
```

Script Output x Task completed in 0.305 seconds

EMPNO	ENAME	SAL
7839	KING	5000
7844	TURNER	1500
7876	ADAMS	1100
7900	JAMES	950
7902	FORD	3000
7934	MILLER	1300

14 rows selected.

Saved: DEMO1~1

Hurricane update Now

Search

Line 86 Column 1 Insert Modified Windows: O

ENG US 10:50 PM 10/17/2025